ELSEVIER Discrete Applied Mathematics 60 (1995) 13–23

DISCRETE APPLIED MATHEMATICS

Work-preserving emulations of shuffle-exchange networks: an analysis of the complex plane diagram

Fred Annexstein, John Franco*

Department of Computer Science, University of Cincinnati, Mail Station 0008, 822 Old Chemistry Bl., Cincinnati, OH 45221-0008, USA

Received 26 November 1991

Abstract

In this paper we show that for each n, the order-n shuffle-exchange network can be emulated by an n-node linear processor array or an n^2-node mesh in a work-preserving manner. An emulation of a computation on a guest network $\mathcal{G}$ is *work-preserving* on a host network $\mathcal{H}$, if the time-processor products are equal to within a constant factor. To achieve this result we demonstrate a uniform many-to-one embedding of the nodes of a shuffle-exchange network into a linear array. We then give a simple, deterministic routing algorithm on the linear array which schedules the communication of messages necessary to achieve the emulation within the required time bounds. The analysis of the algorithm relies on certain statistical properties of the complex plane diagram of the shuffle-exchange network.

1. Introduction

The shuffle-exchange network [8] is one of the most fundamental interconnection networks for parallel computation. It has been demonstrated that a wide range of problems can be solved efficiently on this network. One drawback to this class of networks lies in the engineering difficulties of a large-scale implementation. The complex topology of the graph complicates efficient implementations. Linear arrays and meshes of processors, on the other hand, provide the simplest structure, and yield the most hardware-efficient implementations of large-scale parallel computers. In this paper, we construct an optimal emulation of general computations on a shuffle-exchange network, i.e., in a work-preserving manner, by small size linear arrays and meshes. An *emulation* of a computation of $T_{\mathcal{G}}$ steps on a guest network $\mathcal{G}$ is *work-preserving* on a host network $\mathcal{H}$, if the time required by $\mathcal{H}$ to complete the same computation is bounded by $O(T_{\mathcal{G}}|\mathcal{G}|/|\mathcal{H}|)$. In such an emulation both the guest

*Corresponding author.

SSDI 0166-218X(94)00040-9

$\mathcal{G}$ and the host $\mathcal{H}$ do the same total work, i.e., the time-processor products are equal to within a constant factor. In particular, we discuss work-preserving emulations of 2^n-node shuffle-exchange networks by $\Theta(\sqrt{n})$-node and $\Theta(n)$-node linear arrays. Our technique for achieving work-preserving emulations cannot be extended to $\omega(n)$-node linear arrays because the bisection width of a 2^n-node shuffle-exchange network is $\Omega(2^n/n)$ (see [5, pp. 476–480]).

Our emulations are specified by embeddings of the guest network in the host network, and an algorithm for routing guest-network messages in the host network. Shuffle-exchange networks, linear arrays, and meshes can all be represented as undirected graphs where the nodes are the processors and the edges represent bidirectional communication links between them. We consider embeddings which are many-to-one mappings of the nodes of shuffle-exchange graphs to nodes of the graphs associated with linear arrays and meshes. Such mappings are called *graph embeddings*. In our emulation of a shuffle-exchange network, all messages communicated between a pair of shuffle-exchange nodes must also be communicated between their corresponding image nodes over host graph edges. The routing algorithm schedules these messages using only local control provided by host graph nodes. In this paper we give a graph embedding and a routing algorithm which provide a work-preserving emulation. The graph embedding is built upon the well-known *complex plane diagram* of the shuffle-exchange network [3,6]. The timing analysis of the routing algorithm depends upon certain statistical properties of the complex plane diagram. These properties are derived in the appendix.

2. Shuffle-exchange networks and the complex-plane diagram

We define the *order-n shuffle-exchange network* as the graph of 2^n nodes, each labeled with a distinct bit-string from the set $\{0,1\}^n$. The edges are given as follows: Given $\beta \in \{0,1\}$ and $u \in \{0,1\}^{n-1}$ a *shuffle edge* connects node βu to $u\beta$, and an *exchange edge* connects node $u\beta$ to $u\bar{\beta}$.

To achieve a work-preserving emulation on an n-node linear array, we exploit properties of the *complex-plane diagram* of the shuffle-exchange graph. This diagram locates the nodes of the shuffle-exchange graph as points in the plane. It was first analyzed in [3], and later in [9,6]. The diagram can be described as follows. Let ω_n denote the nth primitive root of unity (that is, $\omega_n = e^{2\pi j/n}$ where $j = \sqrt{-1}$). Given an n-bit binary string $u = a_{n-1} \ldots a_0$ representing a shuffle-exchange node, let ω define the mapping which sends u to the point in the complex plane defined by

$$\omega(u) = a_{n-1}\omega_n^{n-1} + \cdots + a_1\omega_n + a_0.$$

This mapping has some interesting properties. Call a set of nodes which are equivalent up to shuffles a *necklace*, i.e., if we remove all exchange edges from a shuffle-exchange graph, then the connected components of the remaining graph are necklaces. Each necklace maps to uniformly spaced points on a circle in the complex plane with the origin at the center (see Fig. 1). Shuffle edges appear as chords on these

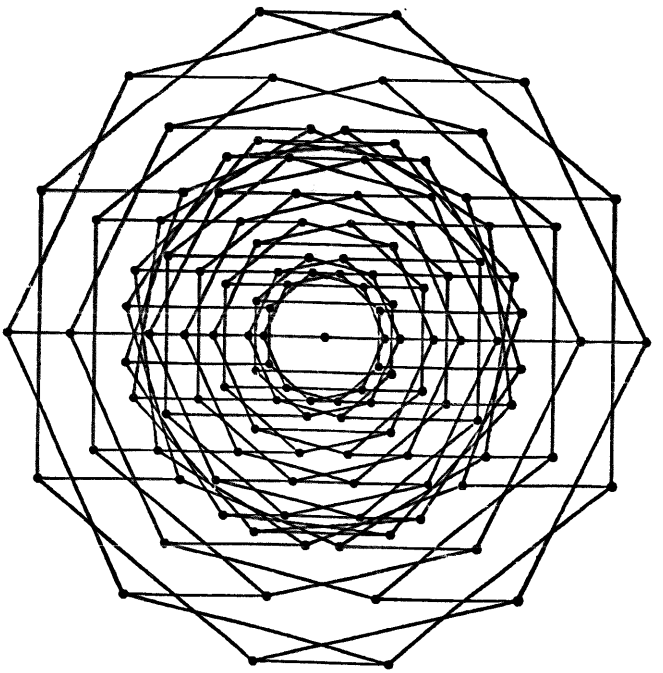

Fig. 1. The complex plane diagram of an order-7 shuffle-exchange network.

circles. To see this, note that each shuffle-edge joins a pair of points in the plane such that one is an ω_n multiple of the other – equivalently, the points differ by a rotation of the plane by $2\pi/n$ radians. Exchange edges can be placed on horizontal line segments of length exactly 1. There are degenerate cases (i.e., necklaces containing less than n nodes) where each node of a necklace is mapped to the origin, but these account for at most $O(2^{n/2})$ of the nodes. Hence, there exist $\Theta(2^n/n)$ non-degenerate necklaces, each containing exactly n nodes (refer to [3,6] for more details).

3. Emulations via graph embeddings

We seek an *emulation* of a generic computation of a guest network $\mathcal{G}$ on a host network $\mathcal{H}$. This is accomplished in part by a *graph embedding*. A graph embedding is specified by

- an *assignment* α of the nodes of $\mathscr{G}$ (in a many-to-one fashion) to the nodes of $\mathscr{H}$,
- a *routing* ρ of each edge (u, v) of $\mathscr{G}$ along a distinct path in $\mathscr{H}$ connecting nodes $\alpha(u)$ and $\alpha(v)$.

The *dilation* of an embedding is the maximum distance in $\mathscr{H}$ between $\alpha(u)$ and $\alpha(v)$, over all pairs u, v of adjacent nodes in $\mathscr{G}$. The *congestion* of an embedding is the maximum number of times any single edge of $\mathscr{H}$ is used by the routing ρ. The *load* of an embedding is the maximum number of nodes of $\mathscr{G}$ mapped to a single node of $\mathscr{H}$ by α. It is clear that, for a given embedding, the maximum of the load, congestion, and dilation is a lower bound on the time (slowdown) for an emulation via that embedding. Somewhat surprisingly, the results of [7] show that an emulation with this slowdown is always possible, up to a constant factor. Their proof is non-constructive in the general case and relies on a randomized algorithm in some special cases of host networks. In this paper we provide a *deterministic* algorithm which will match the lower bound.

It is important that the capabilities of the components of both the host and guest be the same. By this we mean the nodes of $\mathscr{H}$ and $\mathscr{G}$ execute the same set of instructions on words of the same size, and the communication links have the same information carrying capacity. Each $\mathscr{H}$-node must have enough memory to store the states of the set of $\mathscr{G}$-nodes that are assigned to it by α. However, the total memory space of both host and guest are approximately equal.

4. The emulation on linear arrays

4.1. Emulation on a $n^{1/2}$-node linear array

We begin with an easier result that shows how to emulate, in a work-preserving fashion, an order-n shuffle-exchange network $\mathscr{G}$ on a linear array $\mathscr{H}$ of size $O(\sqrt{n})$. Given an n-bit binary string u, define the *weight* of u to be the number of 1-bits it contains. Embed the nodes of the shuffle-exchange graph into the n-node line by mapping each $\mathscr{G}$-node of weight w to the $\mathscr{H}$-node whose index is w. The preimage of maximum size contains $M < 2^n/\sqrt{n}$ nodes. Merge adjacent $\mathscr{H}$-nodes so that eventually each $\mathscr{H}$-node has between M and $2M$ $\mathscr{G}$-nodes mapped to it. What remains of $\mathscr{H}$ is a line of length $O(\sqrt{n})$.

Note that each shuffle edge connects nodes of the same weight. Hence, all shuffle edges connect nodes mapped to the same $\mathscr{H}$-node. All exchange edges connect nodes that are mapped to adjacent $\mathscr{H}$-nodes (or the same $\mathscr{H}$-node).

The routing algorithm is now trivial. In $O(2^n/\sqrt{n})$ time steps deliver messages in which the $\mathscr{G}$-node source/destination pairs are both mapped to the same $\mathscr{H}$-node. Follow this by delivering messages in which the $\mathscr{G}$-node source/destination pairs are mapped to adjacent $\mathscr{H}$-nodes. Since the number of messages is at most $O(2^n)$, and in two steps each pair of adjacent $\mathscr{H}$-nodes can swap messages, the entire routing algorithm takes $O(2^n/\sqrt{n})$ parallel time steps.

4.2. Emulation on an n-node linear array

4.2.1. The graph embedding

Our embedding of the nodes of an order-n shuffle-exchange graph into the n-node line will compose the complex-plane mapping with a projection and a merge mapping. Thus, the embedding is the composition of the following:

1. The complex mapping ω from a bit string $u = a_{n-1} \ldots a_0$ to a point in the complex plane.
2. A projection γ of that point onto the vertical axis. That is, γ maps each point $\omega(u) \mapsto \sum_k a_k \sin(2\pi k/n)$.
3. A merge-of-neighbors mapping μ from the projection to a node in the linear array. This is obtained by partitioning, symmetrically around the origin, the vertical axis into n regions each containing $O(2^n/n)$ projected points. This can be done because there are $O(2^n/n)$ necklaces, so at most $O(2^n/n)$ nodes map to a single point on the vertical axis (degenerate necklaces that map to the origin can safely be ignored). We associate a single node of the n-node line with each element of this partition.

This embedding $\alpha = \mu \circ \gamma \circ \omega$ of the nodes of an order-n shuffle-exchange graph into a n-node line has load $O(2^n/n)$, dilation $O(n)$, and congestion $O(2^n/n)$. The load and dilation bounds follow from the definition of α. The congestion follows from the fact that each necklace contributes at most two edges to the congestion on any edge of the linear array. Thus, the best possible slowdown for the emulation that uses this embedding is $\Omega(2^n/n)$. In what follows, we construct a deterministic routing algorithm which, combined with our embedding, gives a work-preserving emulation with a slowdown matching this lower bound.

4.2.2. The routing algorithm

Here we give a deterministic routing algorithm that finishes within the specified time bound of $O(2^n/n)$. The routing must be orchestrated carefully in order to prevent hotspots. Thus, the algorithm has two separate phases.

Phase 1: Route (internally) all messages whose source and destination are mapped to the same $\mathscr{H}$-node by α.

Phase 2: Route (externally) all messages whose source and destination are mapped to different $\mathscr{H}$-nodes by α. In this phase we will only route messages originating in the same necklace at the same time. Once all messages from a particular necklace are delivered, we move on to route messages from a new necklace. We call each of these a *necklace routing*.

Obviously, there is no edge contention in Phase 1, and hence this phase can be completed in time $O(2^n/n)$. There is no significant edge contention in Phase 2 due to our use of necklace routings: during a necklace routing each host line edge will route a maximum of four messages, since at most two edges of any necklace will congest any host edge. However, it is not obvious that Phase 2 can be completed in $O(2^n/n)$ time since the path lengths of external communication may be very long. In fact, for certain

necklaces that are mapped to points at distances far from the origin, the time necessary to accomplish a necklace routing is $O(n)$.

In the following, we show that the routing time of Phase 2 is $O(2^n/n)$ parallel time steps. The total time to complete Phase 2 is the sum of the times to complete all necklace routings. A necklace routing proceeds until all messages originating from nodes of the necklace finally reach the images of their destinations. Thus, the time for a particular necklace routing is at most the maximum number of host edges between both images of neighboring guest nodes. We can use the following lemma to provide an asymptotic upper bound on this maximum.

Lemma 4.1. *The density function of projected points on the vertical axis converges to a normal distribution having mean* 0 *and variance* $n/4$.

Proof. Follows immediately from Lemma A.2 of the appendix with $t_1 = 1$ and $t_2 = 0$. □

It follows from Lemma 4.1 and well-known properties of the normal distribution than an interval of length l along the vertical axis crosses the maximum number of projected points on the vertical axis in the vicinity of the origin, and this maximum is at most $O(l2^n/\sqrt{n})$. Consider a necklace whose nodes are mapped by ω to points at radius r in the complex plane. The projection of the line segment connecting nearest neighbors in this mapped necklace onto the vertical axis is an interval of length bounded above by $2\pi r/n$. The maximum number of projected points on the vertical axis which this interval crosses is $O((2\pi r/n)(2^n/\sqrt{n}))$. Hence, the maximum number of host line edges, and therefore the time to complete a necklace routing in a necklace mapped to radius r, is bounded from above by $O((2\pi r/n)\sqrt{n})$. The time to do all necklace routings for all necklaces mapped to radius r can be bounded from above by the following expression

$$\Theta\left(\frac{2\pi r}{\sqrt{n}}\right)\sum_i i \Pr(\exists i \text{ necklaces mapped to radius } r). \tag{1}$$

The reader may be surprised by the use of probabilities in the expression above. Although there is no randomness in the placement of nodes in the complex plane, there is, nevertheless, a 0–1 probability distribution describing node locations. We prefer to use this notation for convenience.

Integration of (1) with respect to r yields a bound on the total time over all necklaces, denoted by $T(n)$.

Before integrating (1) we need to find a more usable form of the probabilistic term. This is given by the following lemma.

Lemma 4.2. *For* $r > 0$,

$$\sum_i i \Pr(\exists\, i \text{ necklaces mapped to radius } r) = \frac{2^n}{n} \Pr(\text{random } \mathcal{G}\text{-node maps to radius } r).$$

Proof.

$$\begin{aligned}
&\Pr(\text{random } \mathcal{G}\text{-node maps to radius } r)\\
&= \sum_x \Pr(\text{random } \mathcal{G}\text{-node maps to radius } r \text{ and } \exists x\ \mathcal{G}\text{-nodes mapped to radius } r)\\
&= \sum_x \Pr(\text{random } \mathcal{G}\text{-node maps to radius } r \mid \exists x\ \mathcal{G}\text{-nodes mapped to radius } r)\\
&\quad \times \Pr(\exists x\ \mathcal{G}\text{-nodes mapped to radius } r)\\
&= \sum_i \Pr(\text{random } \mathcal{G}\text{-node maps to radius } r \mid \exists ni\ \mathcal{G}\text{-nodes mapped to radius } r)\\
&\quad \times \Pr(\exists ni\ \mathcal{G}\text{-nodes mapped to radius } r)\\
&= \sum_i \frac{ni}{2^n} \Pr(\exists i \text{ necklaces mapped to radius } r)\\
&= \frac{n}{2^n} \sum_i i \Pr(\exists i \text{ necklaces mapped to radius } r). \quad \square
\end{aligned}$$

Substituting from Lemma 4.2 into (1) and using the integral of the result to bound $T(n)$ gives

$$\begin{aligned}
T(n) &\leqslant \frac{2^n}{n} \int_{r>0} \Theta\left(\frac{2\pi r}{\sqrt{n}}\right) \Pr(\text{random } \mathcal{G}\text{-node maps to radius } r)\,dr\\
&= \Theta\left(\frac{2^n}{n^{3/2}}\right) \int_{r>0} r \Pr(\text{random } \mathcal{G}\text{-node maps to radius } r)\,dr. \qquad (2)
\end{aligned}$$

We can ignore the case when $r = 0$, for all such messages are handled in Phase 1.

From Theorem A.3 in the appendix, we know the probability density function stated in (2). Substituting into (2) gives

$$\begin{aligned}
T(n) &\leqslant \Theta\left(\frac{2^n}{n^{3/2}}\right) \int_{r>0} \frac{r^2}{n} e^{-2r^2/n}\,dr\\
&= \Theta\left(\frac{2^n}{n^{3/2}}\right) \sqrt{n} = \Theta(2^n/n). \qquad (3)
\end{aligned}$$

From (3) we have that Phase 2 takes $O(2^n/n)$ time. Thus, we have the following theorem.

Theorem 4.1. *The embedding α of Section* 4.2.1 *combined with the routing algorithm of Section* 4.2.2 *yields a work-preserving emulation of an order-n shuffle-exchange network by an n-node line.*

5. Conclusion

We have shown that an order-n shuffle-exchange network can be optimally emulated by $\Theta(n)$-node linear arrays in a work-preserving fashion. Our results rely on a new statistical analysis of the complex-plane diagram of the shuffle-exchange networks which may be of independent interest.

Our results are easily extended to achieve a work-preserving emulation of the order-n shuffle-exchange network on an $n \times n$ mesh. This is accomplished using the results above to program the mesh to emulate a direct product of two order-$n/2$ shuffle-exchange networks (it is not difficult to show that an order-n shuffle-exchange network is one-to-one embeddable in such a product graph with dilation 4). Again, by bisection width arguments, it is not possible to achieve work-preserving emulations via graph embeddings to $\omega(n^2)$-node meshes.

Appendix

In this appendix we show that by modeling as a random process, the summing of vectors corresponding to the nth roots of unity, we can obtain limits on the distribution of points in the complex plane diagram of the shuffle-exchange network.

We find the limiting probability density function of the magnitude of the sum

$$\sum_{i=0}^{n-1} e^{(2\pi i/n)j} a_i,$$

where a_i are 0–1 valued independent random variables and j is the complex number $\sqrt{-1}$.

Let $\{V_0, V_1, \ldots, V_{n-1}\}$ be the set of n vectors of unit magnitude with tails at the origin, heads uniformly spaced on the unit circle, and V_0 has no imaginary component. Then $V_i = e^{(2\pi i/n)j}$ and the vertical (respectively, horizontal) component of V_i is $\sin(2\pi i/n)$ (respectively, $\cos(2\pi i/n)$). Let $u = \{a_0, a_1, \ldots, a_{n-1}\}$ be an n-dimensional 0–1 vector selected uniformly from the set of all n-dimensional 0–1 vectors. Let

$$V^{(n)} = \sum_{i=0}^{n-1} V_i \cdot a_i = \sum_{i=0}^{n-1} e^{(2\pi i/n)j} a_i.$$

Denote the vertical (respectively, horizontal) component of $V^{(n)}$ by $V_1^{(n)}$ (respectively, $V_2^{(n)}$). Then

$$V_1^{(n)} = \sum_{i=0}^{n-1} \sin(2\pi i/n) \cdot a_i \tag{4}$$

and

$$V_2^{(n)} = \sum_{i=0}^{n} \cos(2\pi i/n) \cdot a_i. \tag{5}$$

We wish to find the probability distribution of the magnitude of $V^{(n)}$ in the limit as $n \to \infty$. We show that, in the limit, any linear combination of the components of $V^{(n)}$ has distribution which is identical to that of the linear combination, with same multipliers, of the statistically independent, normally distributed components of a random vector. From a result of Cramer and Wold [1] this implies that the distribution of the magnitude of $V^{(n)}$ tends to the distribution of the magnitude of a random vector with normally distributed, independent horizontal and vertical components.

Let $\mu(X)$ denote the mean of the random variable X.

Lemma A.1.

$$\mu(V_1^{(n)}) = \mu(V_2^{(n)}) = 0.$$

Proof. We consider only $V_1^{(n)}$ here, $V_2^{(n)}$ follows similarly.

$$\begin{aligned}\mu(V_1^{(n)}) &= \sum_{i=0}^{n-1} \sin(2\pi i/n)/2 \\ &= \sin(0)/2 + \sum_{i=1}^{\lfloor n/2 \rfloor} \sin(2\pi i/n)/2 + \sum_{i=\lfloor n/2 \rfloor + 1}^{n-1} -\sin(2\pi - 2\pi i/n)/2 \\ &= \begin{cases} \sum_{i=1}^{\lfloor n/2 \rfloor} (\sin(2\pi i/n) - \sin(2\pi i/n))/2 = 0, & \text{if } n \text{ is odd,} \\ \sum_{i=1}^{\lfloor n/2 \rfloor - 1} (\sin(2\pi i/n) - \sin(2\pi i/n))/2 + \sin(\pi) = 0, & \text{if } n \text{ is even.} \end{cases}\end{aligned}$$

This proves the lemma. □

Lemma A.2. *Let t_1 and t_2 be any real numbers. Then the scalar $P = t_1 V_1^{(n)} + t_2 V_2^{(n)}$ is normally distributed with mean $\mu = 0$ and variance $\sigma^2 = (t_1^2 + t_2^2)(n/4)$.*

Proof. From (4) and (5),

$$P = \sum_{i=0}^{n-1} (t_1 \sin(2\pi i/n) + t_2 \cos(2\pi i/n)) a_i = \sum_{i=0}^{n-1} Z_i,$$

where the random variables $Z_i = (t_1 \sin(2\pi i/n) + t_2 \cos(2\pi i/n)) a_i$ take two values in the range $(-(t_1 + t_2), (t_1 + t_2))$ and are mutually independent because all a_i are. Hence, the central limit theorem applies [2] and P is normally distributed. The mean of P is 0 from Lemma A.1 and the fact that the expectation of the sum of random variables is the sum of the expectations of those variables. The variance of P is

$$\sigma^2 = \sum_{i=0}^{n-1} \sigma^2(Z_i).$$

But,

$$\begin{aligned}\sigma^2(Z_i) &= (t_1 \sin(2\pi i/n) + t_2 \cos(2\pi i/n))^2/4 \\ &= (t_1^2 \sin^2(2\pi i/n) + t_2^2 \cos^2(2\pi i/n) + 2t_1 t_2 \sin(2\pi i/n)\cos(2\pi i/n))/2 \\ &= t_1^2(1 - \cos(4\pi i/n))/4 + t_2^2(1 + \cos(4\pi i/n))/4 + 2t_1 t_2 \sin(4\pi i/n)/4.\end{aligned}$$

As in Lemma A.1 it may be shown that

$$\sum_{i=0}^{n-1} \sin(4\pi i/n) = 0 \quad \text{and} \quad \sum_{i=0}^{n-1} \cos(4\pi i/n) = 0.$$

Hence $\sigma^2 = \sum_{i=0}^{n-1} \sigma^2(Z_i) = (t_1^2 + t_2^2)(n/4)$. This proves the lemma. □

In order to show that $V_1^{(n)}$ and $V_2^{(n)}$ are statistically independent in the limit we make use of a consequence of the Cramer–Wold theorem which is stated as follows.

Theorem A.1. *The vector* $(V_1^{(n)}, V_2^{(n)}) \Rightarrow (V_1, V_2)$ *in distribution if and only if*

$$\forall (t_1, t_2) \in \mathscr{R}^2,\ t_1 V_1^{(n)} + t_2 V_2^{(n)} \Rightarrow t_1 V_1 + t_2 V_2$$

in distribution.

Proof. Follows from [1, p. 397]. □

We note that Theorem A.1 relates vector distributions to the distributions of the sums of their components. The fact that vectors are reduced to scalars makes this theorem so useful here.

Let $V = (V_1, V_2)$ be a random vector the components of which are statistically independent with normal distribution having mean 0 and variance $n/4$.

Theorem A.2. *The vector* $(V_1^{(n)}, V_2^{(n)})$ *tends to* (V_1, V_2) *in distribution as n gets large.*

Proof. Since V_1 and V_2 are independent and normally distributed with mean 0 and variance $n/4$, $t_1 V_1 + t_2 V_2$ is normally distributed with mean 0 and variance $t_1^2(n/4) + t_2^2(n/4)$. Thus, from Lemma A.2, $t_1 V_1 + t_2 V_2$ has the same distribution in the limit as P. The hypothesis follows from Theorem A.1. □

Theorem A.2 may be used directly to get the desired result.

Theorem A.3. *The probability distribution of the magnitude of* $V^{(n)}$ *tends toward the density function*

$$\frac{2}{\pi n} e^{-2(x^2+y^2)/n},$$

where x and y are the magnitudes of the horizontal and vertical components.

Proof. From Theorem A.2 the horizontal and vertical components are statistically independent and normally distributed with mean 0 and variance $n/4$ in the limit. Therefore, the limiting density function of the magnitude of $V^{(n)}$ is the product of the density functions of the horizontal and vertical components. The theorem follows. □

Acknowledgments

We wish to thank Joanna Mitro and Magda Peligrad for pointing us to the results of Cramer and Wold.

References

[1] P. Billingsley, Probability and Measure (Wiley, New York, 1986) 397.
[2] W. Feller, An Introduction to Probability Theorey and its Applications, Vol. I (Wiley, New York, 1966).
[3] D. Hoey and C.E. Leiserson, A layout for the shuffle-exchange network, in: Proceedings of the International Conference on Parallel Processing (1980) 329–336.
[4] R. Koch, F.T. Leighton, B. Maggs, S. Rao and A.L. Rosenberg, Work-preserving emulations of fixed-connection networks, in: Proceedings of the 21st ACM Symposium on Theory of Computing (1989) 227–240.
[5] F.T. Leighton, Introduction to Parallel Algorithms and Architectures (Morgan Kaufmann, San Mateo, 1992).
[6] F.T. Leighton, M. Lepley and G.L. Miller, Layouts for the shuffle-exchange graph based on the complex plane diagram. SIAM J. Algebraic Discrete Methods 5 (1984) 202–215.
[7] F.T. Leighton, B. Maggs and S. Rao, Universal packet routing algorithms, in: Proceedings of the 29th IEEE Symposium on Foundations of Computer Science (1988) 256–269.
[8] H. Stone, Parallel processing with the perfect shuffle, IEEE Trans. Comput. 20 (1971) 153–161.
[9] D. Steinberg and M. Rodeh, A layout for the shuffle-exchange network with $O(N^2 \log^{3/2} N)$ area, The Wiezmann Institute and IBM Scientific Center, Israel (1980).